Projet PFA

Mise en production d'un système de création et de lecture automatisée de QCM évolués.

Cahier des charges

Client: Pellegrini François

Responsable pédagogique: Rollet Antoine

Elèves: Kesmi Meryem, Delmas Robin, Song Chen
Bizac Antoine, Cardoso David, Gueye Papa, Nedelec Alexandre

16 décembre 2011

Table des matières

1 Introduction et motivation **4**

2 Analyse de l'existant **5**
I Génération du fichier modèle 5
I.A Lecture du fichier QCM 5
I.A.1 Reconnaissance du format du fichier 5
I.A.2 Calibrage du fichier 5
I.A.3 Affichage du fichier 6
I.B Séléction des cases 6
I.C Sauvegarde des informations séléctionnées 6
II Analyse du QCM 6
II.A Chargement du fichier modèle 6
II.B Lecture du fichier QCM rempli 7
II.B.1 Reconnaissance du format du fichier 7
II.B.2 Calibrage du fichier 7
II.C Recherche des réponses 7
II.D Génération du fichier résultat 7
III AMC, Correction automatisée de formulaires QCM 7
III.A Avantages et inconvénients du logiciel 8
IV Diagramme de séquence 8

3 Modèle du système **10**
I Diagramme de cas d'utilisation 10
II Nouveau diagramme de séquence 11
III Nouveaux diagrammes de classes 13

4 Evolution du système **15**
I Adaptations possible 15
I.A Analyse de champs libres 15
I.B Calcul d'une note 15

5 Nouvelles spécifications **17**
I Génération du fichier modèle 18
I.A Lecture du fichier QCM 18
I.A.1 Reconnaissance du format du fichier 18
I.A.2 Calibrage du fichier 18
I.A.3 Affichage du fichier 19
I.B Sélection des champs 20
I.B.1 Choix du taux de noir pour qu'un(e) case/ratio soit considéré(e) comme rempli(e) 20
I.B.2 Améliorer la détection des cases/ratios noircies . . 20
I.B.3 Choix de l'ordre dans lequel les informations sont inscrites sur le fichier de résultat 22
I.C Sauvegarde des informations sélectionnées 22
II Analyse du QCM 23
II.A Chargement du fichier modèle 23
II.B Lecture du code barre 23
II.C Lecture du fichier QCM rempli 25
II.C.1 Reconnaissance du format du fichier 25
II.C.2 Calibrage du fichier 26
II.D Recherche des réponses 26
II.E Génération du fichier résultat 26
II.E.1 Sauvegarde des résultats de l'analyse du QCM . . . 26
III Génération d'un code barre 27
III.A Portabilité du logiciel 28
III.B Simplification de l'utilisation du logiciel 28
III.B.1 Contrainte 28
III.B.2 Solutions proposées 28
III.C Facilité d'installation 29

6 Diagramme de Gantt **30**

7 Conclusion **32**

8 Lexique **33**

9 Bibliographie/Webographie **35**

Table des figures

2.1 Diagramme de séquence 9

3.1 Diagramme de cas d'utilisation 10
3.2 Diagramme de séquence 12
3.3 Diagrammes de classes 13
3.4 Diagrammes de classes 14

5.1 Contour actif 21
5.2 Délimiter une zone 21
5.3 Clic à l'intérieur de la case 22
5.4 Deux échecs 22
5.5 Balayage du code-barre 24
5.6 Cadre de code-barre de Bordeaux 1 25

6.1 Diagramme de Gantt 30

Chapitre 1

Introduction et motivation

Notre projet est une reprise d'un projet de PFA proposé l'année dernière, qui consistait à créer un logiciel d'analyse de QCM en faisant une étude au préalable des outils et d'analyseurs déjà existants.
Notre travail consistera donc à reprendre le prototype réalisé par le groupe de l'année dernière, et à lui apporter d'importantes modifications dans un but bien précis : le rendre fiable, mature et donc opérationnel sur le marché.

Il est à signaler que le prototype réalisé est bien fonctionnel et intéresse déjà le service des examens de l'Université Bordeaux 1. Les défauts et problèmes rencontrés ainsi que les retours que nous avons eu de la part de Mr Alexandre Moret de ce service même vont faire l'objet de nos interventions. Cette intéraction avec M Moret est d'autant plus importante qu'elle nous permet de mieux comprendre les besoins réels des futurs utilisateurs et d'adapter au mieux notre logiciel pour les satisfaire.

Une étude appronfondie du logiciel en question nous a permis de faire le point sur ses fonctionnalités majeures. En effet, le logiciel a deux principales fonctionnalités : Génération de fichier modèle à partir du QCM en format PDF ,et Analyse du QCM rempli.
Après avoir testé le logciel nous avons aussi pu comprendre son réel fonctionemment. Il s'agit tout d'abord d'insérer le QCM vierge en format PDF, l'interface graphique du logiciel permet de visualiser ce dernier dans un cadre bien délimité, puis d'effectuer la génération du fichier modèle qui va être stocké en format XML. Ensuite, le QCM rempli est chargé à son tour, toujours en format PDF. Un fichier résultat en format CSV est finalement généré après analyse du QCM rempli.

Chapitre 2

Analyse de l'existant

Une étude de l'existant nous a permis de mettre l'accent sur les différents besoins auxquels répond le logiciel. Celui-ci a deux fonctionnalités majeures qui sont : La génération d'un fichier modèle à partir d'un QCM vierge, et l'analyse de QCM remplis.

I Génération du fichier modèle

A ce stade là, on opère sur un fichier QCM vierge en entrée.

I.A Lecture du fichier QCM

I.A.1 Reconnaissance du format du fichier

Le logiciel fonctionne avec des fichiers PDF en format *bitmap*. Ceci dit il reconnait certains PDF en format *vectoriel*.

I.A.2 Calibrage du fichier

Au niveau de la génération du fichier modèle intervient la bibliothèque Pandore, écrite en C++, qui est une bibliothèque de traitement d'image. Elle opère au niveau de la recherche et la reconnaissance de marqueurs présents sur le QCM et qui permettent de corriger les éventuelles rotations que pourrait avoir subies le document lors de la procédure de scan.

Fonctionnalités de Pandore

La bibliothèque Pandore récupère une image du QCM qu'elle convertit en format PAN de codage RGB. Ensuite, elle binarise l'image en utilisant le principe de

seuillage. Enfin, en se basant sur le principe de morphologie mathématique qu'est l'ouverture, Pandore détecte les marqueurs , et ce suite à l'application succesive d'érosion et de dilatation du rayon des marqueurs avec pour élément structurant un cercle.

L'équipe de l'année dernière a choisi de travailler avec quatre marqueurs, tous de forme circulaire, qui est la forme la plus simple à détecter, si l'on considère que le document peut avoir pivoté. Le logiciel ne commence son traitement sur la feuille du QCM qu'une fois qu'elle a été réorientée grâce au calcul des angles d'inclinaisons. Cependant le logiciel n'utilisant que 2 maqueurs, il ne gère pas les problèmes de rotation trop importantes (180° par exemple) ni les changements de résolution.

I.A.3 Affichage du fichier

Une fois le fichier calibré, le logiciel permet de l'afficher dans un encadré bien délimité comme le montre l'interface graphique.

I.B Séléction des cases

La génération du fichier modèle se base sur un certain nombre d'informations obtenues suite à l'étape de délimitation des zones du QCM. Le logiciel permet en effet, par simples clics de la souris, d'enregistrer des zones sous forme de rectangles qui représentent les carrés des radios ou des cases à cocher.

I.C Sauvegarde des informations séléctionnées

C'est à partir de ces informations stockées que l'on génère le fichier modèle. Ceci passe d'abord par la création d'un arbre représentant de manière structurée toutes ces informations qui ne sont autre que le QCM lui même traduit en arborescence. Grâce à la bibliothèque XMLEncoder ,l'arbre est codé en XML,de ce fait on obtient le fichier modèle en format XML.

II Analyse du QCM

II.A Chargement du fichier modèle

Le logiciel permet à l'utilisateur de sélectionner le fichier modèle généré lors de l'étape précédente qui correspondra au fichier QCM rempli à analyser.

II.B Lecture du fichier QCM rempli

II.B.1 Reconnaissance du format du fichier

De manière similaire à précedemment, le logiciel reconnait les fichier PDF en format *bitmap*, et renvoie une erreur le cas échéant (erreur lecture fichier).

II.B.2 Calibrage du fichier

Le calibrage du fichier se fait avec les mêmes marqueurs afin de positionner correctement le QCM rempli.

II.C Recherche des réponses

Afin de récupérer les coordonées des champs à lire, on utilise les coordonnées des champs contenus dans le fichier xml correspondant au fichier modèle généré lors de l'étape précédente. Les coordonnées des cases et autres zones représentent en effet le squelette du QCM, c'est bien l'arborescence des zones délimitées au préalable que l'on retrouve en décodant le fichier XML (fichier modèle) à l'aide d'un algorithme présent dans la bibliothèque XMLDecoder.
Le pourcentage de pixels noirs dans les cases et ratios permet de savoir si une case/ratio est noircie ou non. Une zone analysée est considérée comme noircie si le pourcentage de pixels noirs dans la zone atteint 35%. Ces informations sont par la suite stockées dans un fichier CSV de manière ordonnée.

II.D Génération du fichier résultat

Le fichier CSV est généré au fur et à mesure du parcours de l'arborescence des zones délimitées et dépend de la détection ou non des zones noircies. Ce fichier contient un ensemble de valeurs qui correspondent aux diverses informations liées au QCM (nom de la matière, nombre de pages etc..), mais aussi celles représentant les réponses aux questions, tout cela dans un orde prédéfini.

Il est à noter qu'à chaque élève correspond un fichier CSV, et que les valeurs décrites précédemment sont écrites chacune sur une ligne.

III AMC, Correction automatisée de formulaires QCM

Le client nous a fourni la référence d'un logiciel open source de correction de QCM, le logiciel AMC dont certaines fonctionnalités sont identiques à celles souhaitées par le client pour ce projet. En effet ce logiciel peut prendre en entrée

un fichier PDF pouvant contenir un nombre important de QCM (éventuellement d'un grand nombre de pages) scannés et les analyse en gérant les éventuelles rotations des fichiers grâce à 4 marqueurs ronds situés aux coins des feuilles. Le fichier modèle doit être obligatoirement un fichier latex généré avec ou sans l'aide du logiciel. Les feuilles disposent de vecteurs binaires afin de distinquer les numéros de QCM et de pages, et ont également un champ de cases à noircir par les étudiants afin d'établir leur numéro d'étudiant. Ainsi, AMC présente par de nombreux aspects des similarités avec le logiciel que nous allons implémenter. Une étude plus complète d'AMC sera menée par la suite, nous nous contenterons dans ce cahier des charges de donner et bref aperçu des avantages et inconvénients du logiciel.

III.A Avantages et inconvénients du logiciel

Le logiciel est installable sous différentes versions de Linux et sous MacOS. En revanche, il n'est présentement pas installable sous Windows et ne le sera sans doute pas facilement puisqu'il est implémenté notamment en C++ et en Perl avec différentes bibliothèques extérieures. De ce fait, il n'est pas facilement installable sous ubuntu par exemple car il nécessite l'installation de nombreuses bibliothèque annexes.

AMC dispose de fonctionnalités supplémentaires comme le traitement direct des résultats des QCM analysés. En effet il est possible de corriger directement les QCM sur les fichiers PDF grâce à un barème entré par l'utilisateur. De plus, la création du fichier modèle via latex permet au logiciel de repérer très facilement coordonnées des cases réponses aux questions sans que l'utilisateur n'ait à intervenir. Cependant, la contrainte d'un fichier modèle au format latex est assez importante : l'utilisateur est obligé de créer son fichier modèle sous latex ce qui n'est pas trivial pour la majortié des personnes. Pour les services de l'université des Sciences de Bordeaux par exemple cela ne serait tout simplement pas envisageable car cela demanderait une formation des professeurs au latex.

IV Diagramme de séquence

Ce diagramme décrit l'enchaînement dans le temps des différentes actions effectuées par le créateur des modèles du *QCM* et l'analyseur du *QCM*. Il a été récupéré dans le rapport de pfa du goupe de l'année dernière.

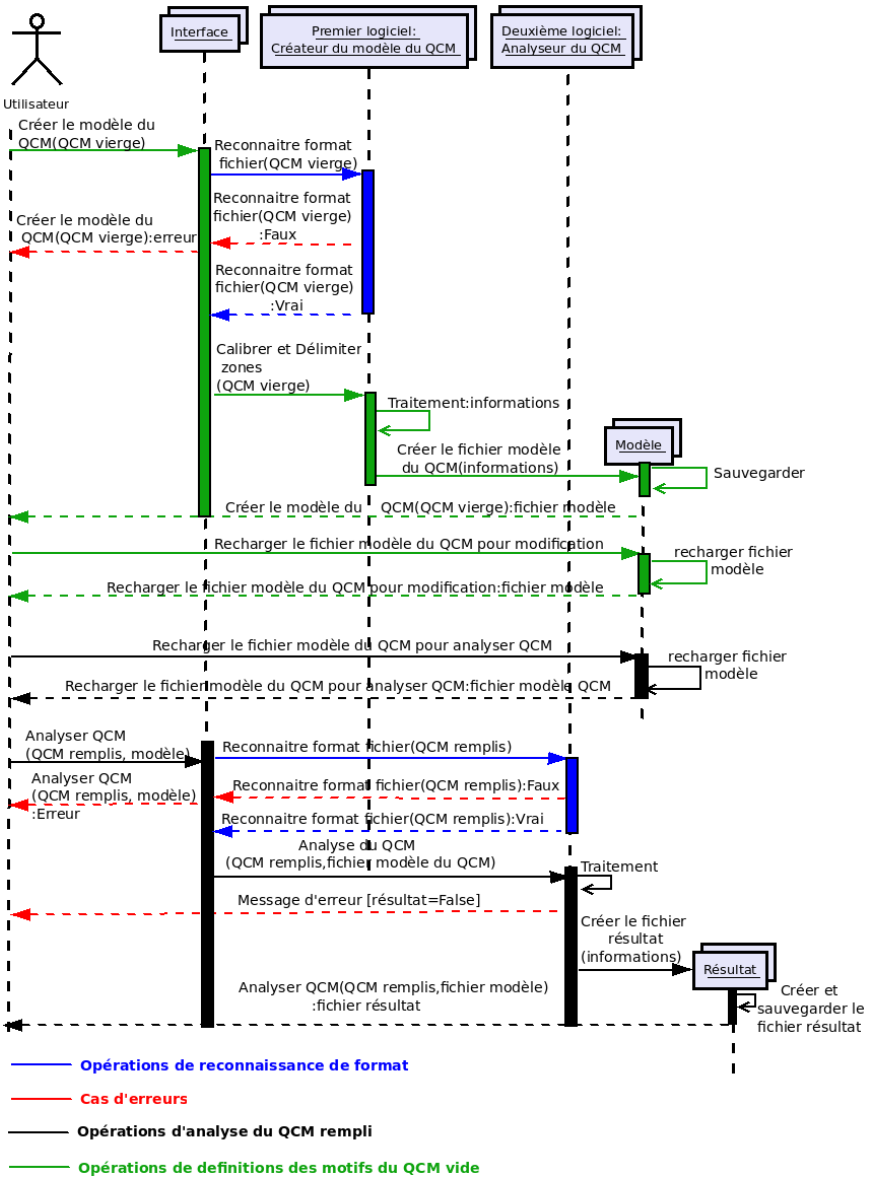
Utilisateur
Interface
Premier logiciel:
Créateur du modèle du QCM
Deuxième logiciel:
Analyseur du QCM
Créer le modèle du QCM(QCM vierge)
Reconnaitre format fichier(QCM vierge)
Reconnaitre format fichier(QCM vierge) :Faux
Créer le modèle du QCM(QCM vierge):erreur
Reconnaitre format fichier(QCM vierge) :Vrai
Calibrer et Délimiter zones (QCM vierge)
Traitement:informations
Modèle
Créer le fichier modèle du QCM(informations)
Sauvegarder
Créer le modèle du QCM(QCM vierge):fichier modèle
Recharger le fichier modèle du QCM pour modification
recharger fichier modèle
Recharger le fichier modèle du QCM pour modification:fichier modèle
Recharger le fichier modèle du QCM pour analyser QCM
recharger fichier modèle
Recharger le fichier modèle du QCM pour analyser QCM:fichier modèle QCM
Analyser QCM (QCM remplis, modèle)
Reconnaitre format fichier(QCM remplis)
Analyser QCM (QCM remplis, modèle) :Erreur
Reconnaitre format fichier(QCM remplis):Faux
Reconnaitre format fichier(QCM remplis):Vrai
Analyse du QCM (QCM remplis,fichier modèle du QCM)
Traitement
Message d'erreur [résultat=False]
Créer le fichier résultat (informations)
Résultat
Analyser QCM(QCM remplis,fichier modèle) :fichier résultat
Créer et sauvegarder le fichier résultat
Opérations de reconnaissance de format
Cas d'erreurs
Opérations d'analyse du QCM rempli
Opérations de definitions des motifs du QCM vide

FIGURE 2.1 – Diagramme de séquence

Chapitre 3

Modèle du système

I Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est exactement le même que celui de la version précédente, et a donc été récupéré du rapport de projet de l'année dernière .

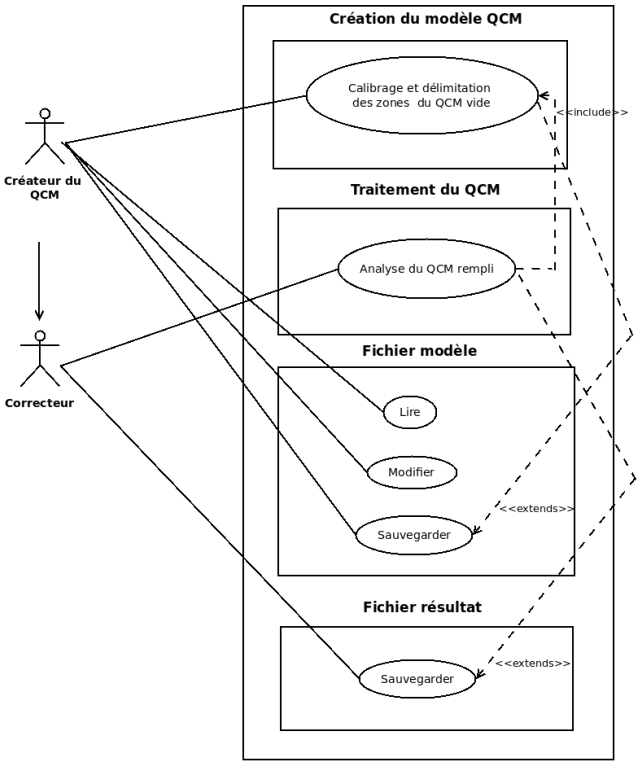

FIGURE 3.1 – Diagramme de cas d'utilisation

II Nouveau diagramme de séquence

Ce diagramme est celui récupéré dans le rapport du groupe de l'année dernière, avec en jaune les fonctionnalités sur lesquelles nous allons intervenir et que nous allons modifier par la suite.

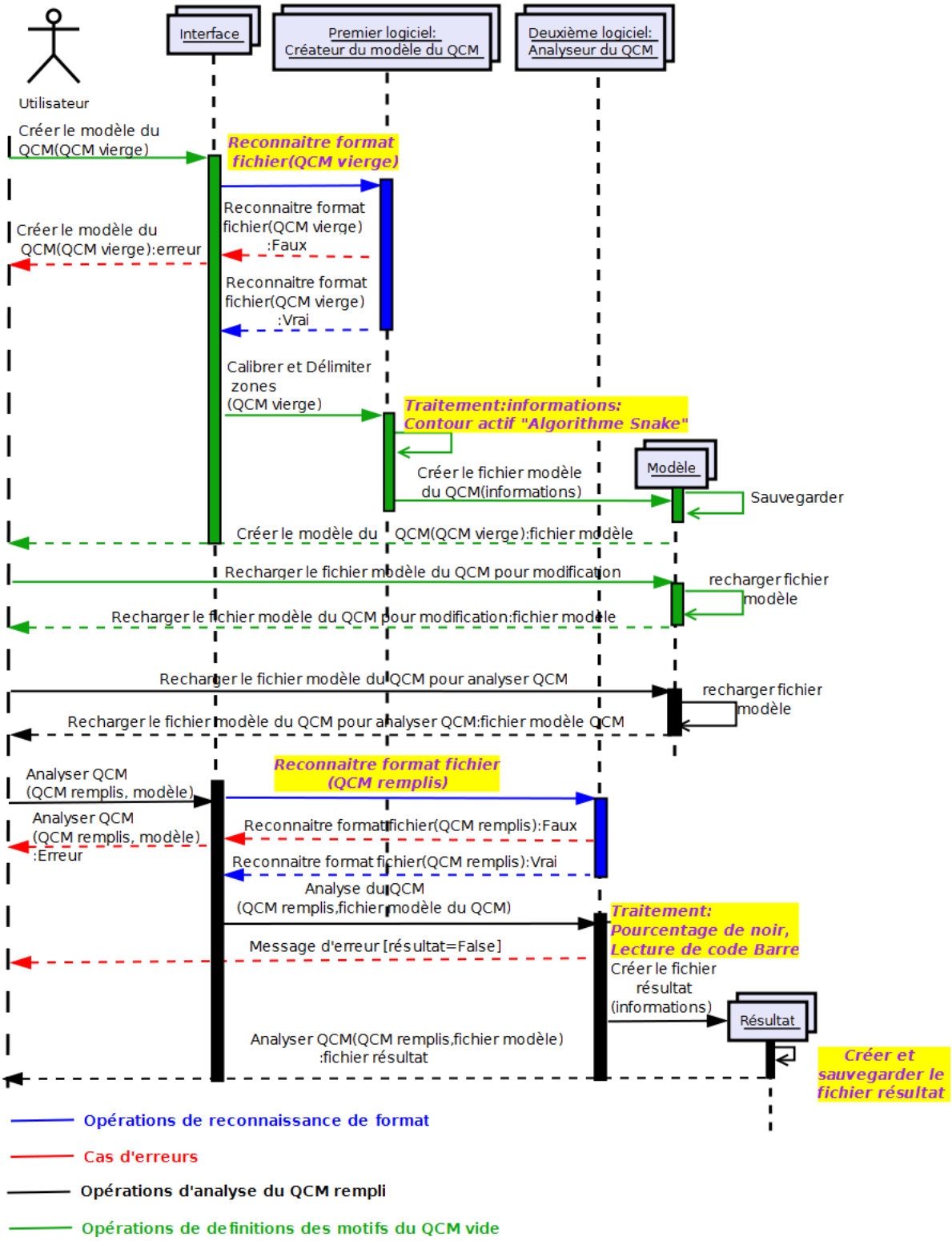
Utilisateur
Interface
Premier logiciel:
Créateur du modèle du QCM
Deuxième logiciel:
Analyseur du QCM
Créer le modèle du QCM(QCM vierge)
Reconnaitre format fichier(QCM vierge)
Reconnaitre format fichier(QCM vierge) :Faux
Créer le modèle du QCM(QCM vierge):erreur
Reconnaitre format fichier(QCM vierge) :Vrai
Calibrer et Délimiter zones (QCM vierge)
Traitement:informations: Contour actif "Algorithme Snake"
Modèle
Créer le fichier modèle du QCM(informations)
Sauvegarder
Créer le modèle du QCM(QCM vierge):fichier modèle
Recharger le fichier modèle du QCM pour modification
recharger fichier modèle
Recharger le fichier modèle du QCM pour modification:fichier modèle
Recharger le fichier modèle du QCM pour analyser QCM
recharger fichier modèle
Recharger le fichier modèle du QCM pour analyser QCM:fichier modèle QCM
Analyser QCM (QCM remplis, modèle)
Reconnaitre format fichier (QCM remplis)
Analyser QCM (QCM remplis, modèle) :Erreur
Reconnaitre format fichier(QCM remplis):Faux
Reconnaitre format fichier(QCM remplis):Vrai
Analyse du QCM (QCM remplis,fichier modèle du QCM)
Traitement: Pourcentage de noir, Lecture de code Barre
Message d'erreur [résultat=False]
Créer le fichier résultat (informations)
Résultat
Analyser QCM(QCM remplis,fichier modèle) :fichier résultat
Créer et sauvegarder le fichier résultat
Opérations de reconnaissance de format
Cas d'erreurs
Opérations d'analyse du QCM rempli
Opérations de definitions des motifs du QCM vide

FIGURE 3.2 – Diagramme de séquence

III Nouveaux diagrammes de classes

Les classes et les flèches en bleu correspondent aux nouvelles fonctionnalités demandées.

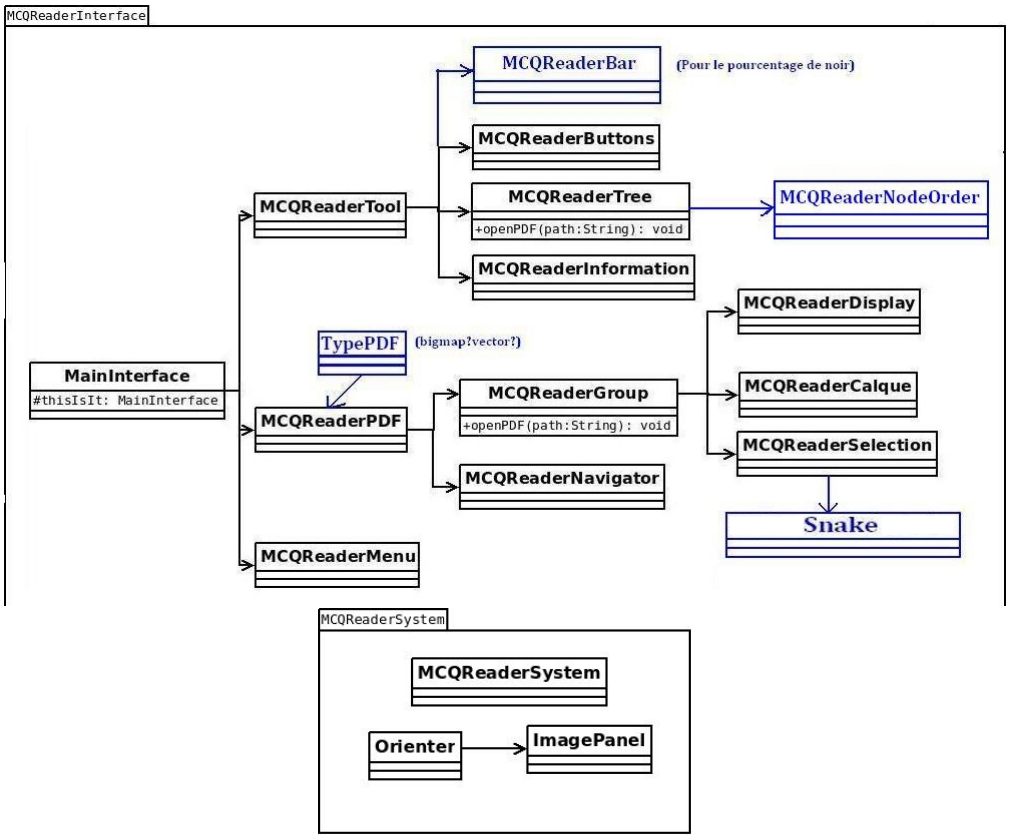

FIGURE 3.3 – Diagrammes de classes

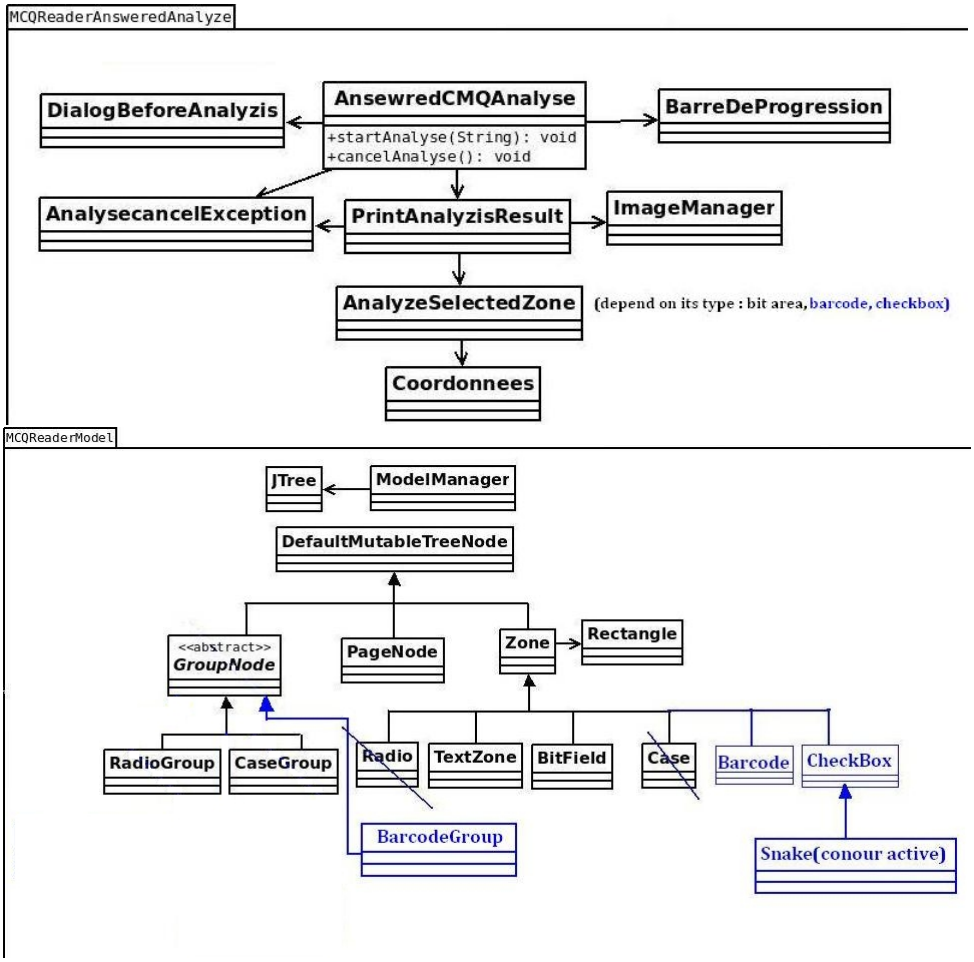

FIGURE 3.4 – Diagrammes de classes

Chapitre 4

Evolution du système

Les QCMs que le logiciel devra analyser sont succeptibles d'évoluer au fil du temps,et pourront contenir de nouvelles informations que l'utilisateur voudrait pouvoir traiter toujours avec un unique logiciel. Celui-ci doit donc être conçu de manière évolutive afin de l'améliorer ou d'ajouter de nouvelles fonctionnalités. Une documentation du code source écrite en Javadoc facilitera cet acte.

I Adaptations possible

I.A Analyse de champs libres

Actuellement, le logiciel prend en compte l'existance de champs libres (zones de texte dans lesquelles l'élève peut écrire une réponse).Cependant même si cette zone en question est selectionnée, elle n'est pas analysée par la suite ce qui a pour consequence la non apparition de cette dernière dans le fichier résultat.
Partant de ce constat,une version ultérieure pourra être prévue afin de permettre à l'utilisateur ayant opté pour la selection de cette zone de voir par exemple dans le fichier résultat une image scanée correspondant à celle delimité par le cadre selectionné lors de la création du fichier modèle.

I.B Calcul d'une note

La version existante du logiciel couplée des nouvelles fonctionnalités implémentées fournira un fichier résultat en csv. A partir de ce fichier, les utilisateurs devront utiliser une macro excel contenant le barême des questions afin d'obtenir une note finale propre à un étudiant. Cependant, certains professeurs qui seront amener à utiliser le logiciel ne sauront pas forcement manier l'outil informatique et sont donc aussi suceptibles de ne pas maitriser excel,encore moins la création de macro. Ainsi, le logiciel pourrait lors de la création du modèle QCM prendre

en compte le barême des questions ainsi que les réponses justes. Dans ce cas là, le logiciel pourrait afficher directement la note finale obtenue par étudiant.

Une autre méthode ou une implémentation complémentaire consiste à l'utilisateur le soin de préciser le barême associé à un QCM lors de la création du modèle de celui-ci. C'est une suggestion d'Alexandre Moret, responsable au service informatique de l'Université Bordeaux 1.

Chapitre 5

Nouvelles spécifications

Dans cette partie, nous présenterons les nouvelles fonctionnalités que devra posséder le logiciel et les contraintes imposées par le client.

Nous commencerons par rappeler les fonctionnalités existantes du logiciel afin de bien comprendre où les nouvelles fonctionnalités vont s'insérer. En bleu figurent les nouveaux besoins fonctionnels et en rouge les besoins fonctionnels qui sont à modifier de manière à inclure de nouvelles fonctionnalités.

I Génération du fichier modèle
  1 Lecture du fichier QCM
    i **Reconnaissance du format du fichier**
    ii **Calibrage du fichier**
    iii Affichage du fichier
  2 **Sélection des champs**
  3 Sauvegarde des informations séléctionnées
II Analyse du QCM
  1 Chargement du fichier modèle
  2 **Lecture du code barre**
  3 Lecture du fichier QCM rempli
    i **Reconnaissance du format du fichier**
    ii **Calibrage du fichier**
    iii Affichage du fichier
  4 Recherche des réponses
  5 Génération du fichier résultat
    i **Sauvegarde des résultats de l'analyse du QCM**
III **Génération d'un code barre**

I Génération du fichier modèle

I.A Lecture du fichier QCM

I.A.1 Reconnaissance du format du fichier

Reconnaissance de fichiers PDF vectoriels

Dans l'optique d'une extensibilité de notre logiciel, sa sphère d'utilisation pourrait être étendue à d'autres traitements en dehors de ceux definis et implémentés dans le code existant.

En effet l'application se limitait pour le moment à lire exclusivement les fichiers *pdf bitmap* grâce à la bibliothèque java Jpedal. Partant de cela une nouvelle fonctionnalité sera mise en place comme stipulée par le client. Cette dernière consistera à prendre en charge la lecture de *pdf vectoriels*.

La bibliothèque java Jpedal ayant évoluée depuis l'année dernière, certains fichiers *pdf vectoriels* peuvent être lus. C'est pourquoi dans un premier temps, une étude sera faite pour vérifier que Jpedal prend en charge tous les types de *pdf vectoriel*. Si ce n'est pas le cas, la lecture de ces fichiers sera mise en place dans un second temps.

Pour se faire une nouvelle bibliothèque sera incorporée à l'ensemble. Elle aura pour rôle de transformer le fichier en question en *pdf bitmap* par le biais de la méthode de *Rasterisation*. Un fichier pdf bitmap sera ainsi créé et sera lui même utilisé par Jpedal.

Intéret de cette methode

- Le code existant ici en l'occurence celui de Jpedal et de ses fonctionalités ne sera pas modifié.
- D'autres types de fichiers pourront être pris en compte dans l'avenir indépendamment du reste.

I.A.2 Calibrage du fichier

Remplacement de la bibliothèqe d'analyse d'image

Nous souhaitons retirer du programme l'utilisation de la bibliothèque Pandore, écrite en C++, qui empêche la portabilité du programme entièrement écrit en JAVA, langage portable par nature. Cette fonctionnalité, fortement liée au besoin

non fonctionnel de portabilité étant une des priorités principales du client, nous avons donc chercher à remplacer Pandore.

Afin de ne plus avoir à utiliser Pandore, nous nous sommes intéressés à la morphologie mathématique de plus près et aux différentes bibliothèques écrites en JAVA qui pourraient contenir de telles fonctionnalités. Nos recherches nous ont menés vers la transformée de Hough. Cette méthode nous permettrait de retrouver dans une image, des éléments proches d'un certain élément constituant, qui peut par exemple être un cercle. C'était intéressant pour nous surtout que nous avions trouvé une implémentation en JAVA de cette transformée. Cette implémentation utilisait la bibliothèque de traitement d'image ImageJ. Ses fonctionnalités en traitement d'images sont très complètes et contiennent des opérateurs de morphologie mathématique comme l'érosion, la dillatation, l'ouverture et la fermeture (dual de l'ouverture). Il nous apparut donc plus simple dans un premier temps d'envisager son utilisation directe remplaçant l'usage de Pandore avec les mêmes fonctionnalités.

Si cela se révélait ne pas convenir pour une raison ou pour une autre, nous pourrions toujours implémenter nous mêmes ces fonctionnalités directement dans notre programme, de façon adaptée à nos besoins.

Pour le moment, le logiciel n'utilise que deux marqueurs et ne pratique que la rotation du document. Contrairement à ce que nous pensions, les deux derniers marqueurs ne sont pas utilisés. Nous comptons donc utiliser un troisième marqueur pour ajuster une déformation éventuelle. Les documents étant scannés ou directement produits depuis un fichier latex, cette fonctionnalité n'est pas primordiale, mais nous essaierons d'en implémenter une solution viable. Nous chercherons à utiliser le quatrième marqueur pour détecter le sens de la page, en le différenciant soit par la forme, soit par le remplissage (les autres marqueurs possédant un centre vide).

I.A.3 Affichage du fichier

Cette partie ne contient pas de nouvelles fonctionnalités, par conséquent le code de cette partie ne sera pas à priori pas modifié.

I.B Sélection des champs

I.B.1 Choix du taux de noir pour qu'un(e) case/ratio soit considéré(e) comme rempli(e)

Cette nouvelle fonctionnalité consiste à ajouter une barre dans le logiciel qui offrira au client la possibilité de choisir un "pourcentage seuil" de noir. Un seuil par défaut sera tout de même présent lors du lancement du logiciel. Toute case insuffisamment noircie, ce qui correspond à un pourcentage de noircissement en dessous du seuil choisi, sera de ce fait considérée comme blanche en d'autres termes non cochée.

I.B.2 Améliorer la détection des cases/ratios noircies

Cette partie consiste à améliorer la détection de case afin de rendre le logiciel plus efficace et plus facile à utiliser.

La méthode qui est expliquée dans la partie précédente n'est pas suffisante pour augmenter l'efficacité de notre logiciel. Si le rectangle choisi est largement plus grand que la vraie case, même avec la possibilité de choisir le taux de noir, le logiciel risque de ne pas fonctionner correctement. C'est pourquoi il est nécessaire d'ajouter une amélioration.

Algorithme "Snake" L'algorithme Snake permet de tracer le contour d'une zone irrégulière en déformant progressivement une courbe de départ.

Grâce à cet algorithme, on est capable de retrouver la frontière d'une case et donc le pourcentage de noir. Pour détecter une case noircie, on peut augmenter drastiquement de le paramètre 35% à environ 80%.

Deux solutions ... A l'aide de l'algorithme de "contour actif", deux solutions sont fournies après une étude réalisée sur les exemplaires de QCM.

- La première solution est d'utiliser le même mécanisme que la méthode existante, puis ajouter à la fin l'application de l'algorithme "snake". C'est-à-dire que le logiciel laisse le client délimiter une zone rectangulaire et puis exécute l'algorithme à l'intérieur de cette zone jusqu'à ce qu'il retrouve la frontière de la forme.
- La deuxième solution consiste pour l'utilisateur à cliquer à intérieur de la case à cocher avant d'appliquer l'olgorithme "snake" de l'intérieur de la case vers l'extérieur.

Leur Avantage ... Chaque solution a son propre avantage. La première solution permet toujours de retrouver ce que l'on veut car il est toujours possible de tracer

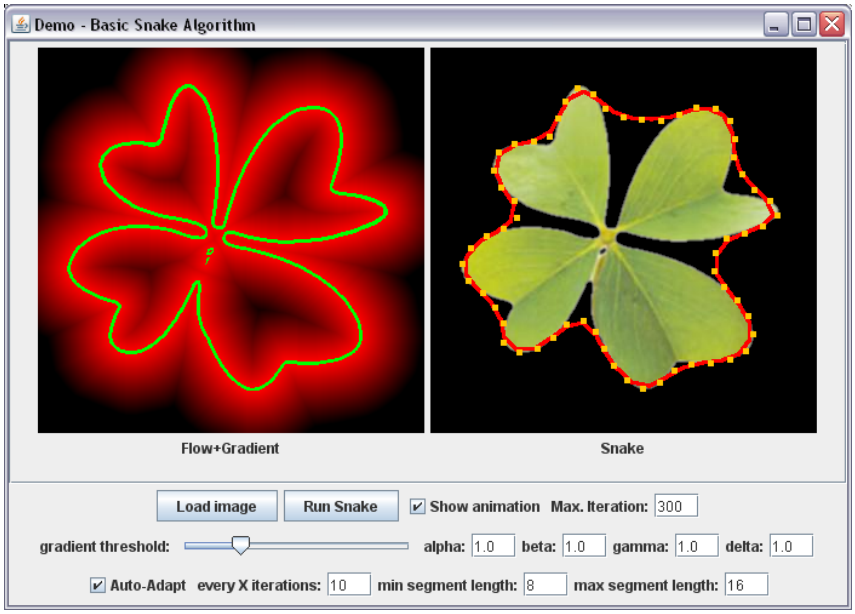

FIGURE 5.1 – Contour actif

FIGURE 5.2 – Délimiter une zone

le rectangle très proche de la vraie forme. Cependant, la seconde solution est plus péférable. Sachant que cliquer dans la case est plus facile à faire et que normalement il n'y a qu'une couleur dans cette zone, elle est plus adaptée aux besoins de client en terme d'utilisation et au niveau de la complexité en temps de l'algorithme (temps d'exécution plus rapide).

Appartition de problèmes Dans les deux cas, il est possible que l'algorithme se trompe.

Puis, il est nécéssaire d'avoir un moyen de mémoriser les coordonnées du dernièr clic de client. La solution que l'on a choisie consite à exécuter instantanément l'algorithme après chaque clic lors de la création du QCM modèle. Le problème affiché dans la figure 5.4 est résolue pour limiter le temps d'exécution de l'algorithme.

FIGURE 5.3 – Clic à l'intérieur de la case

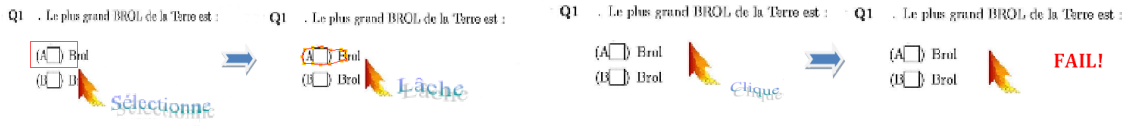
Q1 . Le plus grand BROL de la Terre est :
Sélectionne
Q1 . Le plus grand BROL de la Terre est :
Lâche
Q1 . Le plus grand BROL de la Terre est :
Clique
Q1 . Le plus grand BROL de la Terre est :
FAIL!

FIGURE 5.4 – Deux échecs

I.B.3 Choix de l'ordre dans lequel les informations sont inscrites sur le fichier de résultat

Afin que l'utilisateur puisse obtenir un fichier csv avec les différentes données analysées dans l'ordre qu'il souhaite, l'interface graphique sera modifiée de manière à permettre à l'utilisateur lors de la génération du fichier modèle de visualiser dans la partie gauche du logiciel l'ordre dans lesquels les informations (réponses aux questions,code barre...) apparaitront sur le csv. Il pourra également modifier cette ordre en déplaçant les champs sélectionnés vers le haut ou vers le bas.

De ce fait l'organisation des différents champs dans leur page respective sera supprimée au profit d'une organisation simplement ordonnée par les choix de l'utilisateur.

I.C Sauvegarde des informations sélectionnées

Cette partie ne contient pas de nouvelles fonctionnalités, par conséquent le code de cette partie ne sera pas à priori pas modifié.

II Analyse du QCM

II.A Chargement du fichier modèle

Chargement de plusieurs fichiers modèle

Afin de répondre au besoin cité en II.C.1, il sera nécessaire de pouvoir charger un fichier modèle lors de la création d'un autre fichier modèle. Dans ce cas, seuls les champ de marqueurs binaires (contenant numéro du QCM et numéro de page du QCM) seront conservés.

II.B Lecture du code barre

Afin d'identifier les QCMs remplis par les étudiants, un code-barre propre à chaque étudiant apparaîtra sur le QCM (il faut bien prendre en compte le fait que celui-ci ne sera pas forcément présent sur la première page du QCM), le numéro d'éudiant devra alors être présent dans le fichier csv. En effet, il est nécessaire de pouvoir connaître l'étudiant sans à avoir la copie sous la main.

Le logiciel permet déja à l'utilisateur d'encadrer la zone à étudier en spécifiant la nature de l'objet : un code-barre. Il est donc présent dans l'arbre xml créé. Les actions que nous devons fournir se résument à lire la zone sélectionnée par l'utilisateur contenant le code-barre, à l'analyser et retourner le numéro correspondant au code-barre.

Nous allons voir les 2 méthodes que nous avons retenues pour résoudre le problème. La permière est l'utilisation d'un logiciel externe et la deuxième est la lecture du code-barre par nos propre moyen.

Première méthode

Le logiciel BarCapture permet de lire 4 types de codes-barre (EAN8, EAN13, QRcode et DataMatrix). Développé par la société JaxoSystem, le logiciel est gratuit mais n'est pas open-source. En se mettant en partenariat avec cette entreprise nous allons donc l'utiliser dans le programme pour pouvoir lire des codes-barre.

Le logiciel est programmé en Java. Ainsi, en désassemblant le code source du logiciel, nous pourrons nous infterfacer avec les bibliothèques pour lire le code-barre.

Deuxième méthode

Dans le cas où l'utilisation du logiciel est impossible à cause de l'entreprise, nous devrons programmer nous même la lecture du code-barre.

Plusieurs étapes afin d'obtenir le numéro d'étudiant sont à mettre en oeuvre :
Le code-barre présent sur le QCM peut être incliné, en effet, l'étudiant aura la possibilité de coller l'étiquette du code-barre. Le motif du code-barre étant répétitif, par une transformée de Fournier (Fast Fourier Transform = FFT), nous pouvons déterminer l'inclinaison du code-barre pour pouvoir l'analyser.
Nous effectuons ensuite un balayage horizontal sur le code-barre afin d'obtenir la largeur de chaque barre. A partir de cela, un algorithme se basant sur des calculs arithmétiques simples (additions, multiplications) permet de retrouver le numéro associé à l'étudiant, algorithme qui varie selon le type de code-barre.
Afin d'éviter les erreurs (présence de bruit, tâches...) plusieurs balayages s'effectueront et la largeur d'une barre est calculée à l'aide de la moyenne des balayages.

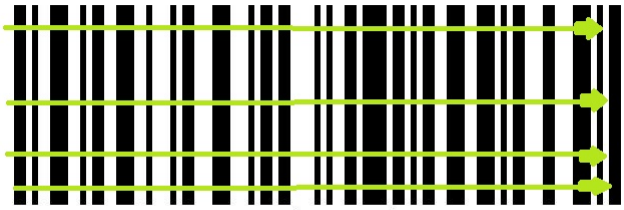

FIGURE 5.5 – Balayage du code-barre

Autre méthode de lecture

Suite a notre rencontre avec la direction informatique de Bordeaux 1, nous avons vu un autre moyen pour reconnaitre l'identifiant d'un étudiant. Il s'avère que, dans les QCms de l'université de Bordeaux 1, le code-barre ne suffit parfois pas. Ainsi, le QCM inclus un module dans lequel il est possible de réécrire son identifiant. Le but est donc de lire les cases de ce module de la même façon que l'on lit les cases du QCM (cf partie reconnaissance des cases).

Voici un exemple de module d'un QCM de Bordeaux 1 :

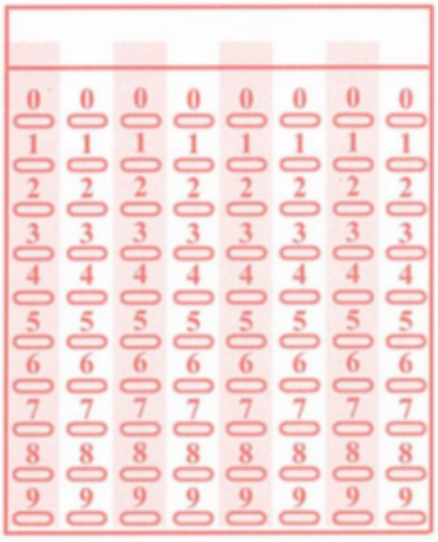

FIGURE 5.6 – Cadre de code-barre de Bordeaux 1

II.C Lecture du fichier QCM rempli

II.C.1 Reconnaissance du format du fichier

Reconnaissance de fichiers contenant plusieurs QCM différents ou non

Il serait fastidieux pour un utilisateur du logiciel de devoir charger chaque QCM scanné au format pdf qu'il veut analyser, d'autant plus que c'est un logiciel qui cible des utilisateurs comme des professeurs d'universités qui ont un très grand nombre d'élèves et donc de QCM à analyser. C'est pourquoi il sera nécessaire de pouvoir charger plusieurs fichiers PDF en même temps pour les analyser ensemble. De même, il doit pouvoir être possible d'analyser un fichier PDF contenant plusieurs QCM. Une demande du client est que ces QCM puissent être différents ie que le logiciel puisse analyser un ensemble de QCM dans un même fichier PDF ou dans des fichiers PDF différents.

De ce fait, il faudra que l'utilisateur puisse sélectionner autant de fichiers modèle différents (.xml) que de fichiers PDF. Pour que le système fonctionne les QCM à analyser ensemble devront tous posséder au même endroit un champ "numéro de questionnaire" (codé par un vecteur de bits), dont la zone aura été repérée précédemment lors de la génération des fichiers modèles. Cela permettra de savoir à quel fichier modèle correspond chaque QCM. A noter que ce champ sera utilisé également pour gérer la numérotation des QCM multi-pages.

II.C.2 Calibrage du fichier

Cette partie ne contient pas de nouvelles fonctionnalités, par conséquent le code de cette partie ne sera pas à priori pas modifié.

II.D Recherche des réponses

Cette partie ne contient pas de nouvelles fonctionnalités, par conséquent le code de cette partie ne sera pas à priori pas modifié.

II.E Génération du fichier résultat

II.E.1 Sauvegarde des résultats de l'analyse du QCM

Une des modifications majeures qui vont être apportées en plus à l'ensemble existant est celle ayant trait à la génération du fichier résultat CSV.
En effet le client a émis le souhait de pouvoir utiliser un tableur qui lui permettra à partir du fichier CSV de générer automatiquement la note des élèves ayant répondu au QCM. Etant donné que la création du tableur est laissée aux soins du client, le CSV en question devra donc respecter un certain nombre de contraintes spécifiées correspondant aux attentes de ce dernier.

Des changements seront ainsi à opérer sur le fichier d'origine.
En voici la liste :

- à chaque QCM analysé correspondra dans le csv une ligne avec tous les éléments sélectionnés par l'utilisateur, et qui permettront la génération du résultat souhaité.
  Par exemple dans le cas d'un fichier de n QCM (possiblement différents) remplis par des élèves on retrouvera les éléments : numéro du QCM, numero du code barre i.e. de l'étudiant, cases cochées... ; et il y aura au total n lignes sur le csv (possiblement plusieurs lignes pour un même étudiant s'il a répondu à plusieurs QCM différents), chaque ligne correspondant à un QCM analysé.
- un fichier type csv sera de la forme :
  $4625 : 5 : 1 : 8 : 0 : 3 : -1 : 2 :$
  $5233 : 6 : 2 : 4 : 2 : 4 : 8 : 3 :$
  $6217 : 8 : 5 : 2 : 9 : 0 : 0 : 0 :$
  Pour un radio buttom on aura -1 si plusieurs cases sont cochées, 0 si aucune case n'est cochée, et la n-ième case du radio-buttom est cochée. Pour une case à cocher simple on aura 0 si aucune case n'est cohée et le nombre obtenu par addition des valeurs binaires des cases

– lors de la délimitation des cases réponses des questions le respect de l'ordre des pages n'est pas primordial, une amélioration qui pourra être apportée sera l'ajout d'une contrainte spécifiant un ordre de parcours croissant du QCM avec une détection automatique de la dernière page traitée.

– l'ordre d'affichage des différentes informations dans une ligne du csv devra correspondre à l'ordre défini par l'utilisateur lors de la sélection des différents champs dans la génération du fichier modèle
Ceci pourra facilement se faire dans la mesure où le csv est créé grâce au parcours de l'arborescence des zones délimitées comme cela est visible dans la partie *Arbre d'affichage des zones délimitées* de notre interface graphique. Il faudra cependant stocker les informations lues lors de l'analyse dans une structure avant de les écrire dans le csv puisque le parcours du PDF ne correspondra pas forcément à l'ordre voulu dans le fichier résultat.

– l'utilisateur pourra choisir un nom de fichier csv dans lequel il veut enregister son fichier résultat, si ce fichier existe déjà les données seront ajoutés à la fin du fichier (en mode append)

III Génération d'un code barre

En marge du développement du logiciel, le client nous a demandé de créer un package Latex, lequel pourra être inclus par les créateurs de QCMs. Celui-ci n'aura qu'à choisir le format de code-barre qu'il souhaite utiliser et de préciser ensuite le code qu'il souhaite insérer dans le code-barre.

Les différents formats que nous allons utiliser sont pour le moment les 3 codes les plus utilisés : EAN 8, EAN 13 et le QRcode. Les 2 premiers sont très utilisée notamment au niveau commercial. Ils permettent de coder respectivement 8 et 13(voire 12) chiffres. (cf annexe pour exemples) Le QRcode permet quant à lui de coder jusqu'à 4296 caractères. Cela permet dans certains cas de préciser de nombreuses informations rien que dans le code-barre. On remarquera cependant que le programme permet de créer de nombreux d'autres codes-barre.

La génération devra pouvoir se faire suivant des options telles que :
– l'inclinaison (seulement un Nombre compris entre 0 et 360)
– la couleur
– la taille

Pour ce qui est de l'implémentation de toutes ces options par le créateur du QCM, si ce dernier ne précise pas les options, chaque option aura des attributs de bases (noir pour la couleur, 0 pour l'inclinaison). Pour ceci, les packages qui vont être créés vont se baser sur un package bien connu de Latex : PSTricks et plus particulièrement pst-barcode. C'est un package créer par Terry Burton et Herbert Voß. (cf Webographie pour une documentation complète en anglais)

Après concertation avec le client, la méthode proposée lui convient et une première documentation lui a été fournie. Une documentation plus détaillée, notamment au niveau des options possibles dans le package, lui sera fournie lors de la remise finale du projet (ou avant).

Besoins non fonctionnels

III.A Portabilité du logiciel

Une des contraintes majeures de ce projet est la portabilité. En effet, il a été spécifié par le client que le logiciel devait pouvoir au moins fonctionner sur les ystèmes d'exploitation Linux, MacOS et Windows. Or présentement il ne fonctionne que sous java. L'implémentation devra donc être effectuée entièrement en java qui était le langage implémentant la première version de ce logiciel. Par ailleurs, comme il l'a été précisé précédemment la blibliothèque C++ Pandore actuellement utilisée par le logiciel devra être remplacée par une bibliothèque java équivalente.

III.B Simplification de l'utilisation du logiciel

III.B.1 Contrainte

Le client a demandé à ce que le logiciel soit simple d'usage ce qui implique qu'il soit relativement ergonomique et qu'un utilisateur ne connaissant pas le logiciel puisse l'utiliser. Or ce n'est pas vraiment le cas actuellement. En effet la première utilisation du logiciel nécessite de disposer du manuel d'utilisation à côté de soi pour pouvoir se servir du logiciel. De plus, ce manuel d'utilisation actuel n'est pas complet et pas totalement clair.

III.B.2 Solutions proposées

Nous ne disposerons probablement pas assez de temps pour rendre l'interface graphique plus ergonomique qu'elle ne l'ait présentement sur les fonctionnalités déjà implémentées. En revanche, nous tacherons d'en tenir compte lors des modifications du code déjà existant et nous chercherons à implémenter la partie graphique

des solutions aux nouvelles fonctionnalités de manière à ce qu'elle soit la plus intuitive possible pour l'utilisateur.

Par ailleurs afin de rendre la prise en main du logiciel plus facile, le manuel d'utilisation sera mis à jour et rendu plus explicite. Il sera également mis en place une aide lors du survol d'un élément de manière à ne pas avoir besoin de toujours se référer à la documentation.

III.C Facilité d'installation

Le logiciel doit pouvoir être installé par un plus grand nombre de personnes, certains usagers pouvant ne pas être informaticiens, il est important de faciliter l'installation du logiciel. Dans ce but une java-archive (.jar) devra être fournie pour chacun des systèmes d'exploitation suivant : Linux, Windows, MacOS.

Eventuellement, si la phase de développement prend de l'avance sur les délais il est envisagé de réaliser pour Windows et MacOS une installation graphique via un assitant d'installation. Tandis que l'utilisation d'un système de gestion de paquets RPM sous Linux pourra compléter la facilité d'installation du logiciel.

Chapitre 6

Diagramme de Gantt

Voici le planning que l'on a prévu pour le déroulement du projet :

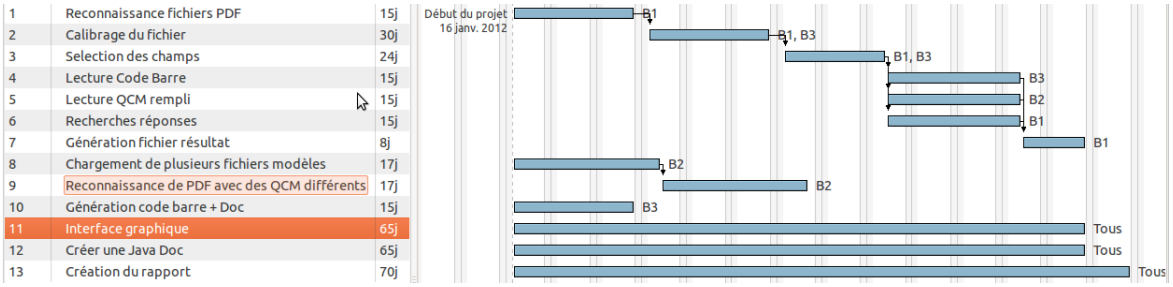

FIGURE 6.1 – Diagramme de Gantt

Ce diagramme montre comment va s'organiser notre projet. Notre équipe sera composée de 3 binômes qui seront chacun assigné a une tâche. Par exemple le binôme 1 va s'occuper de tous ce qui est en rapport avec la bibliothèque de reconnaissance des formes tandis que le binôme 3 travaillera plutôt les parties concernant le code barre.

Bien que nous allons tenter de respecter ce planning, ce diagramme sera a coup sûr modifier, puisqu'il est compliqué d'évaluer dès aujourd'hui la complexité de chaque tâche séparemment. Il sera donc interressant de comparer ce diagramme avec le diagramme final, lequel sera créé au fur et à mesure du projet.

Enfin on remarque que les parties *Interface Graphique, JavaDoc, Rapport* sont supperposées a l'ensemble du projet. Pour ce qui est de l'interface graphique, sachant que chaque modification risque de modifier les fonctionnalités du logiciel, il faudra être réactif et la modifier en conséquance.

Chapitre 7

Conclusion

La première version existante du logiciel ne semble pas aboutie, elle contient de nombreux bug et ne correspond pas totalement aux attentes des utilisateurs ce qui représente un défaut majeur concenant la correction des QCMs. De plus sa première prise en main est loin d'être intuitive ce qui laisse la fade impression d'inachevé. Néanmoins, le travail fourni par l'équipe précédente facilite notre tache dans le fait qu'il souligne les points du logiciel à améliorer ou à retravailler. Ce cahier des charges spécifie les besoins en se basant sur le retour de plusieurs utilisateurs afin de le rendre plus ergonomique, portable et fiable. Nous espérons de plus ajouter des fonctionnalités additionelles afin de simplifier l'utilisation du logiciel, de traiter le fichier csv ou de créér son propre QCM qui s'adaptera au logiciel.

Chapitre 8

Lexique

Codage RGB Pour représenter une couleur, l'un des codages possible est le RGB (Red, Green, Blue). Ce codage consiste à représenter l'espace des couleurs à partir de trois rayonnements monochromatiques de couleurs : rouge (R), vert (G), bleu(B). Cet espace de couleur correspond à la façon dont les toutes les couleurs sont codées informatiquement.

La bibliothèque Pandore commence donc par récupèrer une image du QCM au format GIF qu'elle convertit dans un format propre à Pandore (PAN). Ensuite on extrait de cette image la composante verte. Le résultat correspond à une image en niveau de gris de codage RGB.

Binarisation La binarisation consiste à considérer chaque pixel comme noir ou comme blanc. Ceci est déterminé en définissant un seuil au dessus duquel le pixel est noir et au dessous duquel il est blanc. Cette étape permet un traitement plus simple de l'image et est donc plus rapide.

Ouverture L'ouverture en morphologie mathématique correspond à l'application successive d'une érosion et d'une dilatation, avec un élément structurant.

Dans l'application, pour rechercher les marqueurs on applique ce principe sur l'image avec pour élément structurant un cercle du rayon du marqueur. Il doit résulter de cette opération l'image ne contenant que les marqueurs. Il est alors aisé de repérer leurs centres, permettant ainsi de redresser la feuille du QCM scanné (ces points formant un rectangle dont les côtés sont parallèles au bord de la feuille).

Erosion et Dilatation Pour expliquer simplement, on peut dire que l'érosion réduit tous les "ensembles de pixels" en suivant la forme et la taille de l'élément structurant. Elle permet donc d'effacer à peu près tout ce qui ne ressemble pas à l'élment stucturant. La dilatation est le procédé inverse.

L'application de l'érosion avec l'élément structurant cercle de rayon $R_{marqueurs}$, permet d'effacer tous les éléments autres que les marqueurs. Mais ils auront eux aussi été touchés et en seront ressortis avec une taille réduite. Pour pallier ce problème, on applique une dilatation avec le même élément structurant. Ceci va permettre de récupérer les marqueurs avec leur taille d'origine.

EAN Le code-barre EAN (European Article Numbering) est très utilisé dans le commercial – c'est le code utilisé pour les produits du quotidien. Il permet dans ses 2 versions les plus répendues de coder 8 caractère (resp 13) pour le code EAN8 (resp EAN13).

QRcode Le QRcode (Quick Response Code) est un code qui permet de coder jusqu'à 4296 caractères. Ce code peut contenir toute sorte d'information telles que des URL, du texte, un numéro de téléphone, ... C'est son codage en 2 dimensions grâce à des modules noirs qui permet de coder autant de données.

Chapitre 9

Bibliographie/Webographie

Source pour la bibliothèque de génération de codes-barre pst-barcode :
http://ctan.org/tex-archive/graphics/pstricks/contrib/pst-barcode

Source de BarCapture pour la lecture de codes-barre :
http://www.jaxo-systems.com/solutions/barcapture/

Site du logiciel AMC de correction automatisée de formulaires QCM :
http://home.gna.org/auto-qcm/index.fr